VDM++コード

```
class MaxIndex

operations
  public max_index: seq of nat * nat ==> nat
  max_index(a,n) ==
    (dcl i: nat := 1,
         k: nat := 2;
     while k < n do
       (if a(k) > a(i)
        then i := k;
        k := k + 1);
     return i)
/* VDM++では配列(sequence)のインデックスは１ベース */
  pre
    n > 0 and len a = n
  post
    forall j in set {1,...,n} &
      a(j) <= a(RESULT) and
      (a(j) = a(RESULT) => j >= RESULT);
/* VDM++では副作用のある手続き（操作）の
   検証条件（証明課題）は生成できない */

/* 副作用のある手続きを副作用のない関数に変換
    （while ループは再帰関数に変換） */
functions
  loop: seq of nat * nat * nat * nat -> nat
  loop(a,n,i,k) ==
    if k > n then i
    elseif a(k) > a(i) then loop(a,n,k,k+1)
    else loop(a,n,i,k+1)
  /* ループの不変条件が再帰関数の事前条件になる */
  pre
    n > 0 and len a = n and
    i < k and k <= n+1 and
    forall j in set {1,...,k-1} &
```



```
      a(j) <= a(i) and
      (a(j) = a(i) => j >= i)
  post
    forall j in set {1,...,n} &
      a(j) <= a(RESULT) and
      (a(j) = a(RESULT) => j >= RESULT)
  measure m;
  m: seq of nat * nat * nat * nat -> nat
  m(-,n,-,k) == n-k;

  public f: seq of nat * nat -> nat
  f(a,n) ==
    loop(a,n,1,2)
  pre
    n > 0 and len a = n
  post
    forall j in set {1,...,n} &
      0 < j and j <= n =>
        a(j) <= a(RESULT) and
        (a(j) = a(RESULT) => j >= RESULT);
/* VDM++では再帰関数の検証条件（証明課題）に
   その関数自身が出現するが、
   事前条件・事後条件を展開すれば
   ホーア論理の検証条件と同値になる */
/* VDM++自体は検証条件（証明課題）を
   「生成」するのみで「証明」はしない */

end MaxIndex
```

Coq 用コード

```
Require Import List.

Definition pre (a:list nat) (n:nat) :=
  length a = n /\ n>0.
Definition post (a:list nat) (n:nat) (i:nat) :=
  forall j:nat, j<n ->
    nth j a 0 <= nth i a 0 /\ (nth j a 0 = nth i a 0 -> j>=i).
Definition inv (a:list nat) (n:nat) (i:nat) (k:nat) :=
  pre a n /\
  i<k /\ k<=n /\
  forall j:nat, j<k ->
    nth j a 0 <= nth i a 0 /\ (nth j a 0 = nth i a 0 -> j>=i).

Goal
  (forall a:list nat, forall n i k:nat,
    (pre a n -> inv a n 0 1) /\
    (inv a n i k /\ k<n /\ nth k a 0 > nth i a 0 -> inv a n k (S k)) /\
    (inv a n i k /\ k<n /\ nth k a 0 <= nth i a 0 -> inv a n i (S k)) /\
    (inv a n i k /\ k>=n -> post a n i)).
intuition. (*自動*)

(* 検証条件の１行目 *)
unfold inv.
intuition. (*自動*)

inversion_clear H.
trivial. (*自動*)

inversion_clear H0.
trivial. (*自動*)
inversion_clear H1.

(* 検証条件の２行目 *)
unfold inv.
intuition. (*自動*)
```



```
inversion_clear H0.
trivial. (*自動*)

inversion_clear H0.
intuition. (*自動*)
apply Coq.Arith.Lt.lt_n_Sm_le in H1.
apply Coq.Arith.Lt.le_lt_or_eq in H1.
intuition. (*自動*)
(* j<k の場合 *)
apply H6 in H5.
intuition. (*自動*)
apply Coq.Arith.Lt.lt_le_weak.
revert H2.
revert H1.
apply Coq.Arith.Lt.le_lt_trans.
(* j=k の場合 *)
rewrite H5.
trivial. (*自動*)

(* a[k]が最初の最大要素なので a[j]=a[k]ならば j=k のはず *)
inversion_clear H0.
intuition. (*自動*)
apply Coq.Arith.Lt.lt_n_Sm_le in H1.
apply Coq.Arith.Lt.le_lt_or_eq in H1.
intuition. (*自動*)
(* j<k の場合（矛盾するはず） *)
apply H7 in H6.
intuition. (*自動*)
rewrite H3 in H1.
apply Coq.Arith.Gt.gt_not_le in H2.
apply H2 in H1.
tauto. (*自動*)
(* j=k の場合 *)
rewrite H6.
auto. (*自動*)
```

```
(* 検証条件の 3 行目 *)
unfold inv.
inversion_clear H0.
intuition. (*自動*)

apply Coq.Arith.Lt.lt_n_Sm_le in H4.
apply Coq.Arith.Lt.le_lt_or_eq in H4.
intuition. (*自動*)
(* j<k の場合 *)
apply H5 in H6.
tauto. (*自動*)
(* j=k の場合 *)
rewrite H6.
trivial. (*自動*)

apply Coq.Arith.Lt.lt_n_Sm_le in H4.
apply Coq.Arith.Lt.le_lt_or_eq in H4.
intuition. (*自動*)
(* j<k の場合 *)
apply H5 in H7.
tauto. (*自動*)
(* j=k の場合 *)
rewrite H7.
auto with *. (*自動*)

(* 検証条件の 4 行目 *)
inversion_clear H0.
intuition. (*自動*)
assert (eq : n = k).
auto with *. (*自動*)
rewrite eq.
trivial. (*自動*)
```

CVC3 用コード

```
%%% CVC の配列は無限長（定義域に制限がない）ので length は省略

pre: (ARRAY INT OF INT, INT) -> BOOLEAN =
  LAMBDA (a: ARRAY INT OF INT, n: INT):
    n > 0;

post: (ARRAY INT OF INT, INT, INT) -> BOOLEAN =
  LAMBDA (a: ARRAY INT OF INT, n: INT, i: INT):
    FORALL (j: INT):
      0 <= j AND j < n =>
        a[j] <= a[i] AND
        (a[j] = a[i] => j >= i);

inv: (ARRAY INT OF INT, INT, INT, INT) -> BOOLEAN =
  LAMBDA (a: ARRAY INT OF INT, n: INT, i: INT, k: INT):
    pre(a, n) AND
    i < k AND k <= n AND
    FORALL (j: INT):
      0 <= j AND j < k =>
        a[j] <= a[i] AND
        (a[j] = a[i] => j >= i);

QUERY (FORALL (a: ARRAY INT OF INT, n: INT, i: INT, k: INT):
  (pre(a, n) => inv(a, n, 0, 1)) AND
  (inv(a, n, i, k) AND k < n AND a[k] > a[i] => inv(a, n, k, k + 1)) AND
  (inv(a, n, i, k) AND k < n AND a[k] <= a[i] => inv(a, n, i, k + 1)) AND
  (inv(a, n, i, k) AND k >= n => post(a, n, i)));

%%% COUNTEREXAMPLE;
```


SPIN 用コード

```
#define N 5 /*finite*/

int a[N];
int n = N;
int i, j, k;

init {
  j = 0;
  do
  :: j >= n -> break
  :: j < n && a[j] < 5 /*finite*/ -> a[j] = a[j] + 1
  :: j < n -> j = j + 1
  od;

  do
  :: k >= n -> break
  :: k < n && a[k] > a[i] -> i = k; k = k + 1
  :: k < n && a[k] <= a[i] -> k = k + 1
  od;

  j = 0;
  do
  :: j >= n -> break
  :: j < n && a[j] == a[i] -> assert(j >= i); j = j + 1
  :: j < n && a[j] != a[i] -> assert(a[j] < a[i]); j = j + 1
  od
                                    }
```

BLAST 用コード

```
#include <assert.h>

int max_index(int a[], int n) {
  int i=0, j, k;
  if (n<=0) return -1;
  for (k=1; k<n; k++) {
    if (a[k]>a[i]) {
      i=k;
    }
  }
  for (j=0; j<n; j++) {
    assert(a[j]<=a[i]);
    if (a[j]==a[i]) assert(j>=i);
  }
  return i;
}

int max_index_2(int a0, int a1) {
  int i = 0;
  if (a1>a0) i = 1;

  int ai = (i==0 ? a0 : a1);
  assert(a0<=ai);
  if (a0==ai) assert(0>=i);
  assert(a1<=ai);
  if (a1==ai) assert(1>=i);

  return i;
}

int max_index_3(int a0, int a1, int a2) {
  int i=0, j, k;
  for (k=1; k<3; k++) {
    if (get(a0,a1,a2,k)>get(a0,a1,a2,i)) {
      i=k;
```



```
    }
  }
  int ai = get(a0,a1,a2,i);
  for (j=0; j<3; j++) {
    int aj = get(a0,a1,a2,j);
    assert(aj<=ai);
    if (aj==ai) assert(j>=i);
  }
  return i;
}

int get(int a0, int a1, int a2, int i) {
  switch (i) {
  case 0: return a0;
  case 1: return a1;
  case 2: return a2;
  default: assert(0);
  }
}
```

SLAM（Static Driver Verifier）用コード

```
#include <wdm.h>

#define ERROR {\
  PKSPIN_LOCK sl; PKIRQL il;\
  KeInitializeSpinLock(&sl);\
  KeAcquireSpinLock(&sl,&il);\
  KeAcquireSpinLock(&sl,&il);\
}

int max_index(int a[], int n) {
  int i=0, j, k;
  if (n<=0) return -1;
  for (k=1; k<n; k++) {
    if (a[k]>a[i]) {
      i=k;
    }
  }
  for (j=0; j<n; j++) {
    if (a[j]>a[i]) ERROR;
    if (a[j]==a[i] && j<i) ERROR;
  }
  return i;
}

DRIVER_INITIALIZE DriverEntry;
DRIVER_UNLOAD DriverUnload;

static unsigned int rand() {
  static unsigned int s = 12345;
  s = 22695477 * s + 1;
  return (s / 65536) % 16384;
}

NTSTATUS
DriverEntry
```



```
(IN PDRIVER_OBJECT  DriverObject,
 IN PUNICODE_STRING RegistryPath)
{
  int n, *a, i;

  rand(); rand(); rand();
  n = rand();
  a = ExAllocatePoolWithTag(PagedPool, n*sizeof(int), 0x12345);
  for (i=0; i<n; i++) a[i] = rand();

  max_index(a, n);

  ExFreePool(a);
  return STATUS_SUCCESS;
}

VOID
DriverUnload
(IN PDRIVER_OBJECT DriverObject)
{
}
```